PATENT
HPDNO: 200308580-1

SYSTEM AND METHOD FOR TESTING A MEMORY USING DMA

Background

Computer systems generally include one or more processors and a memory system. The memory system often includes multiple levels of memory devices that range from relatively fast and expensive memory to relatively slow and inexpensive memory. One of the first levels of a memory system is referred to as main memory usually comprises some form of Random Access Memory (RAM). In operation, a computer system loads an operating system and one or more applications into the main memory so that they may be executed by the processor(s).

Because the main memory contains the operating system and applications, it can be a critical component of the computer system. Failures that occur in the main memory can cause broader failures to occur in the system and possibly cause the system to crash. As a result, it is generally desirable to detect errors in the main memory before they cause failures.

Memory errors may be detected by writing known information to a memory and then reading the information back to determine whether it is correct. Some memory errors, however, may be pattern sensitive and may only appear in response to selected information patterns being written to the memory. Some diagnostic testing of a memory may occur in response to a computer system being turned on or reset. This type of testing, however, may not detect errors in computer systems that are left on and not reset for extended periods of time.

Although some memory devices include error correction features that work during operation of a computer system, these features typically detect errors only in response to a specific memory location being read. Because many areas of a memory may not be read with regularity, errors that occur in these areas may go undetected until an access to a faulty memory location takes place.

Accordingly, it would be desirable to be able to detect errors in all areas of a main memory of a computer system before the errors cause failures to occur during operation of the system.

Summary

According to one exemplary embodiment, a computer system is provided that includes a processor, a first bus coupled to the processor, a memory controller coupled to the first bus, a memory coupled to the memory controller, a first input/output (I/O) controller coupled to the first bus, and a test module coupled to the first I/O controller. The test module is configured to cause tests to be performed on the memory using the first bus.

Brief Description of the Drawings

Figure 1 is a block diagram illustrating an embodiment of a computer system configured to test a memory during operation using DMA.

Figure 2 is a block diagram illustrating an embodiment of portions of the computer system shown in Figure 1.

Figure 3 is a flow chart illustrating an embodiment of a method for testing a memory during operation of a computer system using DMA.

Figure 4 is a block diagram illustrating an alternative embodiment of a computer system configured to test a memory during operation using DMA.

Detailed Description

In the following detailed description of the preferred embodiments, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific embodiments in which the invention may be practiced. It is to be understood that other embodiments may be utilized and structural or logical changes may be made without departing from the scope of the present invention. The following detailed description, therefore, is not to be taken in a limiting sense, and the scope of the present invention is defined by the appended claims.

In one aspect of the present disclosure, a computer system includes a test module coupled to an input/output (I/O) controller. The test module is configured to test a portion of the main memory of the computer system during operation of the computer system, i.e., subsequent to the operating system being

booted and during execution of the operating system. The test module provides test transactions to the I/O controller which causes the tests to be performed on the main memory using direct memory access (DMA). The test module may cause remedial action to be taken in response to errors being detected during the tests.

Figure 1 is a block diagram illustrating an embodiment of a computer system 100 configured to test a memory during operation using direct memory access (DMA). Computer system 100 may be any type of computer system such as a handheld, desktop, notebook, mobile, workstation, or server computer. Computer system 100 includes processors 110a through 110(*n*), a core electronics complex 120, a memory 130, and a set of input / output (I/O) devices 140. Processors 110a through 110(*n*) are each coupled to core electronics complex 120 using a set of bus connections 152. Bus connections 152 comprise a set of system busses. Core electronics complex 120 is coupled to memory 130 and I/O devices 140 using connections 154 and 156, respectively. Core electronics complex 120 may also be referred to as a chipset.

Computer system 100 includes any number of processors 110 greater than or equal to one. As used herein, 'processor 110' refers to any one of processors 110a through 110(*n*), and 'processors 110' refers to the set of processors 110a through 110(*n*).

Processor 110a is coupled to a cache 112, and processor 110b includes a cache 114. Caches 112 and 114 may store any type of information such as instructions and data. Other processors 110 may include or be operable with any type or number of caches.

Computer system 100 also includes an operating system (not shown) that is executable by one or more of processors 110. In response to being turned on or reset, one or more of processors 110 cause the operating system to be booted and executed. Processors 110 execute instructions from the operating system and other programs using memory 130.

Core electronics complex 120 includes a memory controller 122, a set of I/O controllers 124, a bus bridge 126, and test module 128. Memory controller 122 is configured to store information into and read information from memory

130 in response to write and read transactions, respectively, from processors 110, test module 128, and I/O devices 140. Memory controller 122 may include hardware and / or software configured to perform memory scrubbing or other error correction functions on memory 130 in response to reading information from memory 130.

I/O controllers 124 may include any type and number of controllers configured to manage one or more I/O devices 140. Examples of I/O controllers 124 include IDE/ATA controllers, SATA controllers, PCI controllers, SCSI controllers, USB controllers, IEEE 1394 (Firewire) controllers, PCMCIA controllers, parallel port controllers, and serial port controllers. In one embodiment, I/O controllers 124 comprise multiple microchips that include an intermediate bus coupled to bus bridge 126, PCI controllers coupled to the intermediate bus, and SCSI, IDE and others controllers coupled to the PCI controllers.

Memory 130 comprises any type of memory managed by memory controller 122 such as RAM, SRAM, DRAM, SDRAM, and DDR SDRAM. In response to commands from system firmware (not shown) or the operating system, memory controller 130 may cause information to be loaded from an I/O device 140 such as a hard drive or a CD-ROM drive into memory 130.

I/O devices 140 may include any type and number of devices configured to communicate with computer system 100 using I/O controllers 124. Each I/O device 140 may be internal or external to computer system 100 and may couple to an expansion slot that is in turn coupled to an I/O controller 124. I/O devices 140 may include a network device (not shown) configured to allow computer system 100 to communicate with other computer systems and a storage device (not shown) configured to store information.

Test module 128 is configured to cause tests to be performed on memory 130 during operation of computer system using DMA by generating test transactions and providing the test transactions to one of I/O controllers 124. The test transactions include read and write transactions configured to cause information to be read from and written to memory 130, respectively. The test transactions may include pattern tests or any other type or combination of read

and write transactions configured to verify the operation and functionality of selected locations in memory 130.

Computer system 100 includes multiple DMA channels that allow test module 128 and I/O devices 140 to interact with memory 130. To use DMA, test module 128 or an I/O device 140 sends a DMA transaction, such as a test transaction, to I/O controller 124. The DMA transaction goes from I/O controller 124 to bus bridge 126 then to memory controller 122 using a bus connection 152. Memory controller 122 causes the DMA transaction to be performed.

Test module 128 may be tailored to operate in conjunction with memory controller 122 depending on the level of error correction functionality that is included in memory controller 122. For example, in embodiments where memory controller 122 implements memory scrubbing or other error correction mechanisms, test module 128 may be configured to primary generate read transactions. In response to the read transactions, the memory scrubbing or error correction functions of memory controller 122 detect and, if possible, fix errors in memory 130. In embodiments where memory controller 122 does not include memory scrubbing or other error correction mechanisms, test module 128 may be configured to write patterns to memory 130 and read back the patterns from memory 130 to detect errors.

The operation of test module 128 will now be described with reference to Figures 2 and 3. Figure 2 is a block diagram illustrating an embodiment of portions of computer system 100, and Figure 3 is a flow chart illustrating an embodiment of a method for testing memory 130 during operation of computer system 100 using DMA. The method is performed by test module 128.

Test module 128 selects and obtains access to a portion 202 of memory 130 during operation of computer system 100 as indicated in a block 302. Depending on the implementation, test module 128 may obtain access to portion 202 by sending a request to the operating system, by sending a request to memory controller 122, by allocating an unused portion of memory 130, or by any other suitable method. In embodiments where memory controller 122 implements memory scrubbing or other error correction mechanisms, test

module 128 may omit the step of obtaining access to a portion of memory 130 before testing the portion.

Test module 128 generates a test transaction as indicated in a block 304. The test transaction may be a read or write transaction that is configured to cause information to be read from or written to portion 202 in memory 130, respectively. Test module 128 provides the test transaction to memory 130 using DMA as indicated in a block 306. In the embodiment of Figure 2, test module 128 accomplishes this by providing the test transaction to an I/O controller 124 using a connection 210. I/O controller 124 provides the transaction to bus bridge 126 using a connection 212. Bus bridge 126 provides the transaction to memory controller 122 using a bus connection 214. Bus connection 214 comprises one of the bus connections 152. Memory controller 122 causes the test transaction to be performed by reading information from or storing information to portion 202 in memory 130. In the case of a read transaction, memory controller 122 causes information to be read from portion 202 and provided to test module 128 using connections 214, 212, and 210.

A determination is made by test module 128 as to whether an error has been detected in portion 202 as indicated in a block 308. If test module 128 detects an error in portion 202, then remedial action is performed as indicated in a block 310. Remedial action may include fixing errors, logging addresses of errors, and / or notifying the operating system, memory controller 122, and / or a system administrator of the errors.

If test module 128 does not detect an error in portion 202, then a determination is made by test module 128 as to whether there are more tests to perform on portion 202 as indicated in a block 312. If there are more tests to perform on portion 202, then the function in block 304 is repeated.

If there are no more tests to perform on portion 202, then test module 128 surrenders access to portion 202 as indicated in a block 314. The function of block 314 may be omitted in embodiments where test module 128 omits the step of obtaining access to portions of memory 130.

A determination is made by test module 128 as to whether there is another portion in memory 130 to test as indicated in a block 316. If there is

another portion in memory 130 to test, then the function of block 302 is repeated. If there is not another portion in memory 130 to test, then the method ends.

In embodiments where memory controller 122 is configured to perform memory scrubbing or other error correction functions in response to reading information from memory 130, errors may be detected and remedial action may be performed by memory controller 122 in response to the test transactions from test module 128. The remedial action may include fixing and logging the errors.

In certain embodiments, computer system 100 includes software diagnostics that can communicate with test module 128. In conjunction with the diagnostics, test module 128 may perform targeted read, write, and pattern testing of memory 130. In response to obtaining access to a portion of memory 130, test module 128 may test specific locations in memory 130 through any combination of reads, writes and pattern testing in response to communications from the diagnostics. The diagnostics may be included as part of the operating system or other applications or may be separate programs.

Test module 128 may be designed to allow a system administrator to configure tests to memory 130 using the operating system or another suitable interface. The operating system or interface may allow the administrator to select, for example, the time and frequency of tests, the particular portions of memory to test (e.g. frequently used portions), and the actions to be taken in the event of failures.

Test module 128 may comprise any suitable combination of hardware and software components.

Figure 4 is a block diagram illustrating an alternative embodiment of computer system 100 configured to test memory 130 during operation using DMA. In the embodiment shown in Figure 4, core electronics complex 420 replaces core electronics complex 120. Core electronics complex 420 includes a system controller 422, I/O controllers 424, and test module 128. System controller 422 includes a memory controller 428.

Test module 128 operates in substantially the same way in the embodiment of Figure 4 as it did in the embodiments of Figure 1, 2, and 3. In

the embodiment of Figure 4, DMA transactions are performed using one of a set of connections 430 between an I/O controller 424 and system controller 422 instead of using a bus connection 152. Accordingly, the test transactions of test module 128 go from test module 128 to an I/O controller 424 to system controller 422 using a connection 430. Memory controller 428 processes the test transactions and, in the case of read transactions, provides information associated with the read transactions back to the I/O controller 424 using connection 430.

As illustrated by Figure 4, test module 128 is configured to operate with different arrangements of core electronic components or chipsets. Accordingly, test module 128 is also configured to operate with other core electronic components or chipsets not illustrated herein.

The above embodiments may provide advantages over systems that do not use DMA to detect memory errors. For example, a computer system may more easily correct memory errors detected using DMA than with other types of memory error detection. In addition, DMA accesses may require less processor overhead than operating system-level processes. Further, operating system-level processes may have recovery problems associated with the operating system.

Although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that a variety of alternate and/or equivalent implementations may be substituted for the specific embodiments shown and described without departing from the scope of the present invention. This application is intended to cover any adaptations or variations of the specific embodiments discussed herein. Therefore, it is intended that this invention be limited only by the claims and the equivalents thereof.